REMARKS

This Amendment is submitted in response to the Examiner's Action mailed April 13, 2004, with a shortened statutory period of three months set to expire July 13, 2004. With this amendment, claims 1, 2, 5-8, 10-14, and 17-19 have been amended.

The Examiner rejected claims 1-19 under 35 U.S.C. § 112, second paragraph, as being indefinite. Specifically, with regard to claims 1, 8, and 13, the Examiner stated that it is not clear what a patch class instruction is. Applicants have amended the claims to refer to a "particular type of unconditional instruction". With regard to claims 2 and 14, the Examiner stated that is it not clear how the first instruction is re-fetched subsequent to overwriting to a second instruction. Applicants have amended these claims to remove this description. Therefore, these rejections are believed to be overcome.

Applicants have amended the claims to describe a first value being written into a memory location. The first value represents a first instruction that is a particular type of unconditional instruction. The first instruction is fetched and executed. While the first instruction is being executed, the first value is overwritten by writing a second value into the memory location. The second value represents a second instruction that is the same particular type of unconditional instruction as the first instruction. Examples of support for these amendments can be found in the specification on page 12, lines 10-21. The memory location is overwritten, while the first instruction is being executed, without producing unexpected results. One example of support for these amendments can be found in the specification on page 12, lines 22-25.

Applicants have amended dependent claims to describe the particular type of instruction being a no-operation type of instruction. Thus, in this case the first instruction that is represented by the first value that is being overwritten is a no-operation type of instruction while the second instruction is also a no-operation type of instruction.

Applicants have amended dependent claims to describe the particular type of instruction being an unconditional branch type of instruction. In this case the first instruction that is represented by the first value that is being overwritten is an unconditional branch type of instruction while the second instruction is also an unconditional branch type of instruction.

Applicants have amended dependent claims to describe the particular type of instruction being only either a no-operation type of instruction or an unconditional branch type of instruction. Thus, in this case the first instruction that is represented by the first value that is being overwritten is only either a no-operation type of instruction or an unconditional branch type of instruction while the second instruction is also only either a no-operation type of instruction or an unconditional branch type of instruction.

Applicants have amended dependent claims to describe the first instruction being executed by a first thread in a simultaneous multiprocessing (SMP) system that executes multiple different threads concurrently utilizing a plurality of processors, and the first instruction being modified by a second thread in the SMP system concurrently with the first thread executing the first instruction. Examples of support for these amendments can be found in the specification on page 10, lines 16-19 and page 11, lines 26-29.

The Examiner rejected claims 1-19 under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent 5,781,776 issued to *Johnston*. This rejection, as it might be applied to the claims as amended, is respectfully traversed.

Applicants' claims are believed to be patentably distinct over *Johnston* because *Johnston* does not describe, teach, or suggest overwriting a second value in a memory location while a first instruction is being executed, the first and second instructions being the same type of instruction, the first and second instructions being unconditional instructions, the first and second instructions both being unconditional branch types of instructions, an SMP system that executes multiple different threads concurrently utilizing a plurality of processors, or a first instruction being modified by a second thread in the SMP system concurrently with the first thread executing the first instruction.

Johnston teaches a method for editing a program that permits changes to be made while the program is executing. An instruction in the program is replaced with a conditional jump instruction.

Johnston does not teach overwriting a second value while the first instruction is executing. *Johnston* teaches replacing an instruction while the program is executing. *Johnston* does not teach replacing an instruction while that particular instruction is executing.

Johnston does not teach a first value that represents a first instruction that is a particular type of instruction and the second value used during overwriting being the same type of instruction. *Johnston* teaches changing an instruction into a conditional jump instruction but does not teach whether the original instruction was also a conditional jump instruction.

Applicants claim the particular type of instruction is an unconditional type of instruction. *Johnston* teaches the replacement instruction is a conditional jump that is executed only if a flag is set. *Johnston* does not teach the replacement instruction being an unconditional type of instruction.

Applicants have amended the dependent claims to describe a first thread in an SMP system executing the first instruction and a second thread in the SMP system modifying the first instruction while the first thread is executing the first instruction. The SMP system executes multiple threads concurrently utilizing a plurality of processors. *Johnston* does not describe, teach, or suggest these features.

Applicants have amended the claims to describe the particular type of instruction being an unconditional branch type of instruction, a no-operation type of instruction, or only either an unconditional branch type of instruction or a no-operation type of instruction. Thus, according to Applicants' claims, the first instruction is either an unconditional branch type of instruction, a no-operation type of instruction, or only either an unconditional branch or no-operation type of instruction. *Johnston* teaches a conditional jump instruction being used as the modifying instruction. *Johnston* does not teach the instruction being modified as being a conditional branch instruction.

Applicants' claims are believed to be patentably distinct over the prior art because the prior art does not describe, teach, or suggest overwriting a second value in a memory location while a first instruction is being executed, the first and second instructions being the same type of instruction, the first and second instructions being unconditional instructions, the first and second instructions both being unconditional branch types of instructions, an SMP system that executes multiple different threads concurrently utilizing a plurality of processors, or a first instruction being modified by a second thread in the SMP system concurrently with the first thread executing the first instruction.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 07.12 04

Respectfully submitted,

[illegible]

Lisa L.B. Yociss
Reg. No. 36,975
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants